Graphical Smalltalk with My Optimization System for Urban Planning Tasks

Graphical Smalltalk with My Optimization System for Urban Planning Tasks

Reinhard Koenig[1], Lukas Treyer[2], Gerhard Schmitt
ETH Zurich, Chair of Information Architecture
http://www.ia.arch.ethz.ch/
[1]reinhard.koenig@arch.ethz.ch, [2]lukastreyer@student.ethz.ch

Abstract. *Based on the description of a conceptual framework for the representation of planning problems on various scales, we introduce an evolutionary design optimization system. This system is exemplified by means of the generation of street networks with locally defined properties for centrality. We show three different scenarios for planning requirements and evaluate the resulting structures with respect to the requirements of our framework. Finally the potentials and challenges of the presented approach are discussed in detail.*
Keywords. *Design optimization; interactive planning support system; generative system integration; evolutionary multi-criteria optimization.*

MOTIVATION

For many computer scientists the programming language Smalltalk was the most pioneering human-computer interaction language of the 1970s. It was designed to be so simple that even children could program. It is one of the first totally object oriented languages – everything is an object. While today many ideas from Smalltalk have since been adopted by other languages, the visionary thinking of the time when it was developed can still inspire us today to strive for flawless human-computer interaction in the development of design optimization systems for architecture and urban planning.

PROBLEM STATEMENT

A number of promising generative algorithms are available today, but none are currently employed to enhance and simplify the day-to-day work of urban planners. Computer support for urban planning projects is usually restricted to basic CAD drawing tools. In the authors' opinion, one reason for the lack of integration of generative methods in planning processes is their complicated handling. Typically they require extensive input of abstract technical rules and parameters that are unfamiliar and daunting for planners.

The situation is further complicated by the fact that planning projects typically consist of a mixture of contradicting and non-contradicting criteria as well as of directly measurable criteria and only indirectly interpretable measures. The lack of suitable optimization methods hinders a systematic evaluation of possible compromises between contradicting planning requirements.

STATE OF THE ART

In their seminal book, Radford and Gero (1988) show various examples of how optimization strategies can be used to solve design problems. Although today we can use more flexible evolutionary optimization methods (Deb, 2001), the concept for their applica-

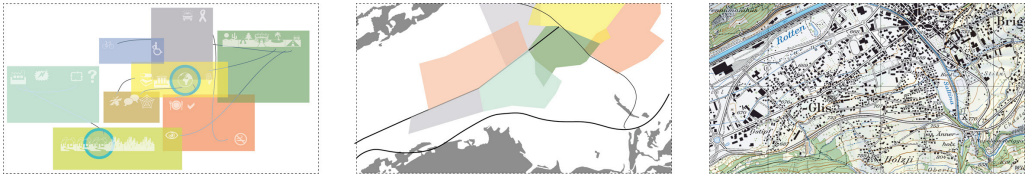

Figure 1
Planning scenario divided into three levels of abstraction. Left: Topological relations between elements and basic properties. Centre: Geometric distribution of the elements. Right: Geometric representation of a possible planning solution [6].

tion and the role of pareto-optimal fronts has not changed a lot over the past few decades (Bentley and Corne, 2002). A good example for state of the art interactive generative planning systems is the work of Derix (2009). Dillenburger et al. (2009) have also presented an interesting system for creating building designs using a weighted-sum optimization algorithm.

Current commercial solutions for generative or procedural modeling, for example Grasshopper [1], GenerativeComponents [2], or CityEngine [3] exemplify the problems with such systems: they require intensive training before they can be used efficiently and though sometimes attractively designed, their user interfaces are not intuitive for urban planners. Furthermore it is not efficient to couple them with optimization tools, because of the increased computing time and restricted possibilities offered by their corresponding APIs. Although Galapagos [4] provides an optimization method for Grasshopper, Rutton (2010) notice that it is only useful for simple problems.

AIMS

Our main goal is to use graphical objects to represent a planning problem and to control an optimization algorithm using primarily these objects. A further challenge is to translate the planner's partially vague qualitative requirements into a precise quantifiable problem representation for an algorithm. Translation problems are one reason why planners rarely embrace computer support. To improve this situation we aim to develop an interactive system for supporting the urban planning process with a more constructive and intuitive interface for planners. The combination of well-designed interaction strategies and planner-friendly problem representation as a basis for evolutionary optimization strategies is as an issue that is yet to be resolved.

CONCEPTUAL FRAMEWORK

To address the aforementioned problems, our first task is to develop a conceptual framework that includes a combination of various interaction strategies for the user interface, different generative techniques, and some optimization methods. We have approached this concept from two perspectives: from that of a planner and from that of a software developer.

To meet the planner's requirements we separate the problem representation and the definition of requirements by at least two levels of abstraction (Figure 1): The first holds the topological relations between various elements and basic properties (Figure 1 left). The graphical objects on this level can encode parameter values e.g. by their size, position or colour, etc. The second abstraction level comprises the geometric representation of possible planning solutions (Figure 1 right). One can interact with all the graphical objects of a current planning proposal on each level to test different options and to refine a planning iteratively. From the software developers point of view we develop a framework for combining evolutionary optimization techniques. These include generative algorithms and evaluation mechanisms to analyze the generated variations. As a basis for this framework, we use state-of-the-art evolutionary multi-criterion optimization methods. For a comprehensive and easily understandable introduction to evolutionary algorithms, see Bentley and Corne (2002). In the following description, we focus only on the essential aspects that are necessary for our purposes. We take the AForge.Net Framework [7] as the starting point for the implementation of

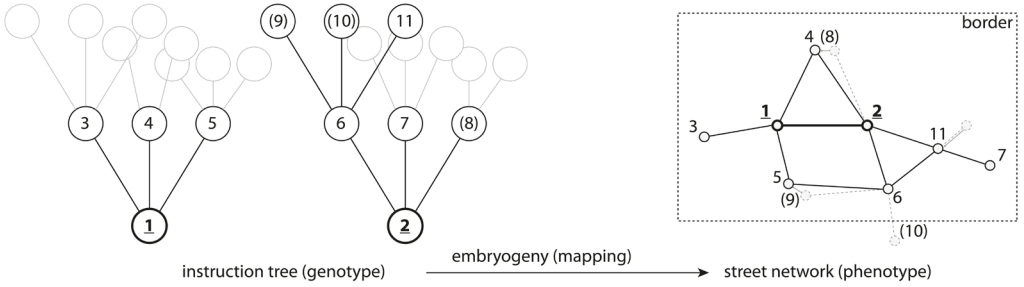

Figure 2
Mapping process from an instruction tree to a street network. The grey dotted street segments on the right side illustrate the adaption of instructions how to add a new segment to a existing network.

our evolutionary algorithm (EA). The main argument for EA is their flexibility in dealing with the problem representation, which is crucial for the problems described above.

From the user's perspective we reverse the logic of generative planning systems: instead of exploring the results of different parameter settings or procedural rule sets, we allow a planner to graphically define what performance a solution shall have and the optimization system automatically generates a set of best compromise solutions. This constellation of deducing a solution from its desired properties is called an inverse problem [5], which was used with a different intention by Koltsova et al. (2012). Based on this concept we develop a method that can be called bi- or multi-directional planning, since one can control a computer-based planning system from any of the abstraction levels as shown in Figure 1.

IMPLEMENTATION

In this paper we present just one part of the aforementioned framework: the optimization of a street network inside a planning area with specific local properties. The sub-areas can be defined by a user with the help of graphical objects in a similar way as shown in Figure 1.

Before we consider some example applications of the optimization system, we first describe the implementation of its basic generative and evaluation mechanisms. Taking the AforgeNet Framework [7] as our starting point, we extend it by a class for a chromosome with mutation and crossover methods and a class for a customized fitness function as described below.

Generation Mechanism

The basic idea for the generative mechanism is to use an instruction tree, which holds the instructions on how to grow a street network (Figure 2). This growth process can be denoted as embryogeny and is responsible for unfolding the abstract information stored in a genotype to the concrete structure of a phenotype, which is why this process can also be described as mapping from genotype to phenotype. For the genotype representation we use a chromosome which in turn is represented in our case by an instruction tree as shown in Figure 2. The rules to create the street network (Figure 2 right) are adapted from the concept of self-sensitive L-Systems by Parish and Müller (2001).

As a basic component of the instruction tree we implement a class for an individual instruction node which stores the information on how to add a new street segment to an existing node of the network (Figure 3). The instructions for a node are reduced to three which is the minimum necessary for our basic system: the range *li* for the length of a street segment, the angle *αi* that indicates the angular deviation from a regular division, and the range *κi* of possible arms at a crossroad. Figure 3 illustrates the mapping of the instructions of three instruction nodes to a phenotype representation. We use the additional parameter tree depth to restrict the size of an instruction tree to a certain limit. The individuals of an initial generation of a population start with randomly assigned instruction values.

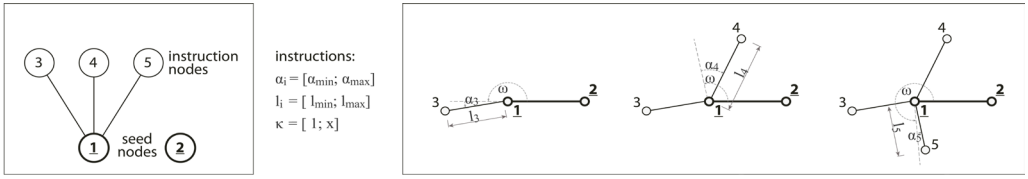

Figure 3
On the left, the tree shows the main instructions of an instruction node. In the box on the right one can see how the instructions are assigned to a street network.

The instruction tree can be mutated and used for crossover operations as illustrated in Figure 4. Since the instructions of a node are always relative to the existing street network, new combinations after the crossover always work. The main reason why we use a tree structure for the chromosome is that it ensures that after the crossover and mutation operations, the corresponding street network remains connected (if the initial network was connected). The mutation operator simply takes (e.g. 1-10%) individual nodes of an instruction tree and assigns a randomly generated value to one of its parameters. The frequency of the execution of these operators at one iteration (or generation) is defined by the crossover and mutation rate.

One of the most important properties of a generative mechanism, as part of an optimization process, is its ability to generate very different network topologies. It is this property that allows an optimization system to find interesting and surprising solutions for a given set of restrictions and goal functions.

Evaluation Mechanism

As a goal function (or fitness function) for the evaluation of the generated street network we use the betweenness centrality (choice) of a network. For this we need to calculate the all-pair shortest paths for the network, and compute this following the concept elaborated by Hochberg [8] using a parallel GPU implementation of the Floyd-Warshall algorithm to calculate shortest paths. For the weightings in the corresponding graph we use angular distances instead of metric distances as introduced by Turner (2001; 2007). The choice value for a specific street segment equals the number of shortest paths from all street segments to all others that pass through that segment. For the sake of simplicity, we use only the choice value in the following examples to characterize street networks, but other centrality measures would be useful too.

EXAMPLE SCENARIO

As a starting situation for the following examples we use an area with the dimensions 3000m × 2000m that needs to be filled with streets (Figure 5). The positions of the existing street connections are marked by nodes with underlined numbers. The areas defined in the right-hand image in Figure 5 will be used in later examples to define a central sub-area (red, dashed) and a quiet sub-area (blue, checked). The red center is placed near the coordinates

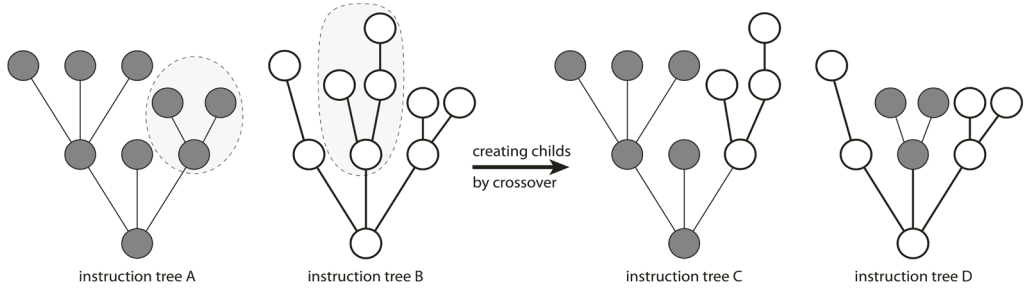

Figure 4
Creation of new child variants (C and D) by a crossover operation applied to two parent instruction trees (A and B).

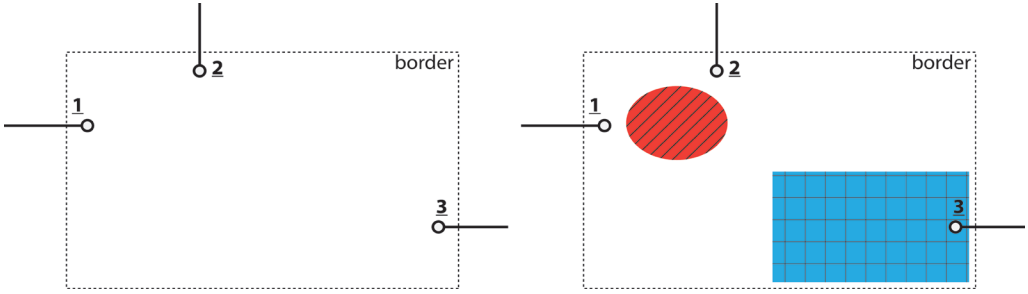

Figure 5
Initial planning situation: The border defines the planning area, and connections to the existing street network are represented by the nodes with underlined numbers. The coloured areas in the right-hand image denote areas where the new street network will have defined properties.

750m/1500m (coordinate origin in the left bottom corner) while the blue quiet area fills the bottom right-hand quarter of our planning area. For the following examples we use the following initial parameters: generations = 50, population size = 50, mutation rate = 0.25, crossover rate = 0.75, tree depth = 8. It is import to select a tree depth high enough to ensure that the complete area can be filled with streets and that there is no indirect restriction for the optimization algorithm. The values for the nodes of the initial instruction trees were initialized with random values with the intervals: *ai* =[-10, 10], *li* =[10, 40], *κi* =[1, 4]

First we consider a simple basic example scenario, where we use the aforementioned optimization method to maximize the maximum choice value (Figure 6) and the sum of all choice values of the generated street network (Figure 7). We start with an empty area as shown in Figure 5 on the left. Figure 6 and Figure 7 show three resulting street networks with the corresponding diagram of the development of the fitness values.

To achieve very high choice values, the most obvious strategy is to design a network, that is separated into two parts which are connected by the most used street, the one with the highest choice value. In cities we find this situations, for example, in places where a bridge crosses a river or a narrow valley divides a settlement. If we consider the three street networks in Figure 6 we can see this structure in the network in the right-hand image. Of the three networks in Figure 6, however, the network in the middle has the best fitness value, although there are no two separate parts. This results from the fact that for the calculation of the trips we use the shortest angular distance and not the shortest metric distance. Because of this, there is one street segment at the top-center which is used very often. If we look at the

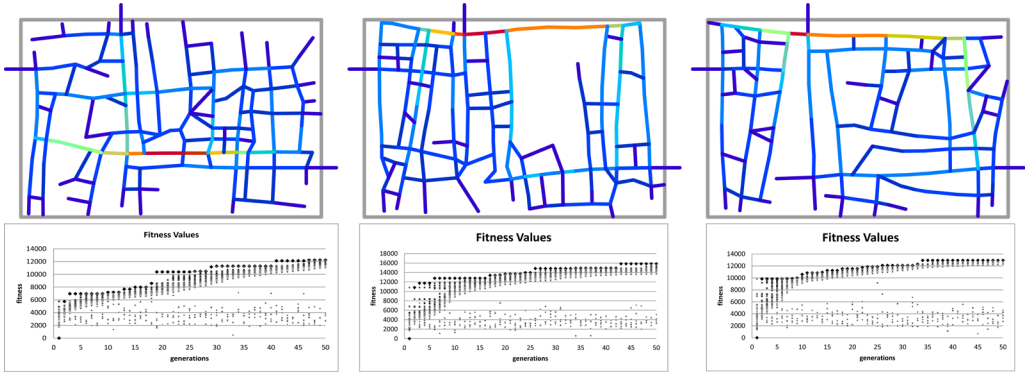

Figure 6
First example scenario. For each of the street networks shown, the maximum choice value is maximized. Red represents street segments with high choice values and blue low choice values. The diagrams in the bottom row show the development of the fitness values over 50 generations.

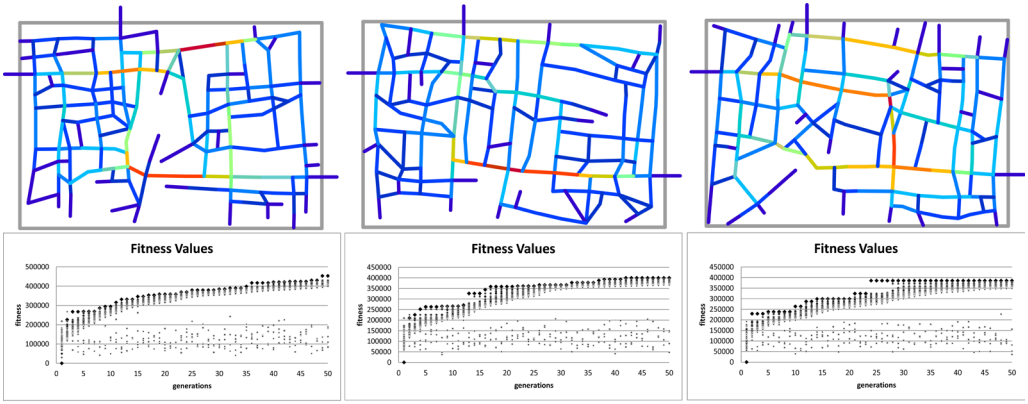

Figure 7
Second example scenario: For each of the street networks shown, the sum of all choice values is maximized. Red represents street segments with high choice values and blue low choice values. The diagrams in the bottom row show the development of the fitness values over 50 generations.

left-hand network in Figure 6 we find that there is no single bottleneck, and at least one other populated route. As a result this network has the worst fitness of the three.

The fact that the three best networks have different maximal fitness values (diagrams in the bottom row of Figures 6 and 7), indicates that the evolutionary optimization process explores different parts of the search space each time it is run. But despite the small differences in the maximum fitness values of the variants, they all fulfill the requirements relatively well. When we consider the random points (representing randomly generated variants) we can clearly see the advantage of using the evolutionary search process compared to randomly generated solutions. The best variants are improved continuously over the 50 generations and reach a level, which cannot be achieved by a random generation process.

In our second example we use the same initial scenario as in the first one, but we adapt the fitness function to maximize the sum of all choice values of the street network. The topologies of the resulting networks in Figure 7 are clearly different to those in Figure 6. Here we cannot see separate network parts and the streets segments with the highest choice values are not concentrated at one location but distributed across the network. This difference proves that our optimization system is working as expected. The development of the fitness shown in the diagrams in the bottom row in Figure 7 is similar to that of the corresponding diagrams in Figure 6. This indicates that both fitness functions direct the search process in a similarly efficient way.

Our third example is based on the initial scenario with two defined sub-areas as shown in the right-hand image of Figure 5. In Figure 8 the central sub-area is shown as a dotted ellipse and the quiet sub-area as a dotted rectangle. To include the spatial aspect in the fitness function, we have to define how to represent the graphical objects that represent the sub-areas with the corresponding specified properties.

First we consider the central sub-area. To achieve a highly-populated center in the defined sub-area we want to locate the street segment with the maximized choice value in it. Therefore we measure the distance *dcmax* of the street segment with the maximum choice value *cmax* to the center. This distance can be used as a weight so that we can decrease the fitness of a network according to the distance *dcmax*:

$$F1 = c_{max} * \left(1 - (d_{cmax}/D)\right)^2, \quad (1)$$

where *D* is a constant which denotes the maximal possible distance. In our examples, this is the diagonal of the border rectangle *D* = 3606m.

Secondly, we consider the quiet sub-area. To achieve an area with as little traffic as possible, e.g.

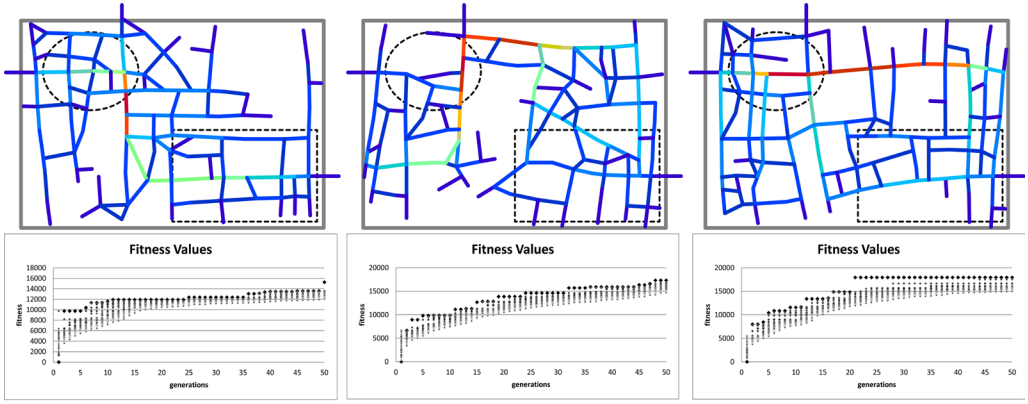

Figure 8
Third example scenario. For each of the street networks shown, we use a combined fitness function: one factor is the maximized maximum choice value that is weighted with the distance to the central area (dotted ellipse), while the second factor is the average sum of all choice values which are assigned to street segments inside the quiet area (dotted rectangle). Red represents street segments with high choice values and blue low choice values. The diagrams in the bottom row show the development of the fitness values over 50 generations.

for residential usage, we want to have only streets with low choice values in it. Therefore we sum all differences of the choice values from the street segment *ci* that are located inside the quiet area *A* and the maximum choice value of the network *cmax*. The average of this sum is used as the second part of our fitness function:

$$\mathbf{F2} = \left(\sum_{i=1}^{n}\left(c_{max} - c_i^A\right)\right)/n \quad (2)$$

From the two fitness values *F1* and *F2* we calculate the final fitness value as the sum of both:

Fitness = *F1* + *F2*. (3)

The street networks resulting from this optimization process are shown in the top row of Figure 8. The results fulfill both of our requirements relatively well: the red coloured street segments with maximum choice values are located close to or inside the central sub-area, while we find primarily only blue coloured street segments with low choice values in the quiet sub-area. To evaluate the effect of the defined sub-areas on the resulting street networks we can compare the variants from Figure 6 and Figure 7 with the ones in Figure 8. We can observe very different structures in comparison to the second example scenario in Figure 7 and similar structures to our first example scenario in Figure 6 showing the two separated network parts (Figure 8 in the middle). In general the results seems self-evident, but nevertheless we can see some problems, e.g. at the solution in the left-hand image of Figure 8. Here the maximum choice value is at the edge of the central area and there is a relatively populated road in the quiet sub-area. Maybe the optimization algorithm could have found a better solution with more generations. But the combined fitness function may be hindering the improvement of this variant. We will discuss this problem in the next section.

DISCUSSION

As outlined in the description of our framework, we have demonstrated a method of representing planning requirements using graphical objects that can be used by an optimization system (Figure 5). The main challenge of the system is interacting with design variants, not because of the complicated user interface – it is, for example, possible to change the genotype and thus the later optimization process by manipulating the graphical objects of the phenotype (street segments and crossroads). This makes it possible to realise a multi-directional planning method as described above.

The main problem of our system is that the optimization process is much too slow for use in an interactive process. The computation of the above examples needed 2030 minutes on an average modern notebook. One generation therefore needed half a minute: half a second would be a more acceptable timespan. Of course these times depend a lot

on the size of the street network, the population size and other aspects. But the main time-critical aspect is the computation of the all-pair shortest path. This needs to be improved in future using an optimized algorithm and more powerful hardware.

In addition we use a very inefficient method to generate instruction trees. We use a random initial process which produces a very huge tree from which only a small fraction of nodes are needed to grow the street network. In the examples above, in the case of $\kappa i = 4$, we have 3^depth instruction nodes for each tree. For a tree depth of 8 this results in max. 3^8 = 6561 nodes, but we only have approximately 300 street segments. Alternatively one could create random but meaningful street networks in the beginning and encode them to make much more efficient instruction trees.

Variations of the angles and placement of the initial street segments as shown in Figure 5 have a relatively significant impact on the further growth and thus on the final phenotype of the network. Therefore, to search for optimal solutions it may also be useful to vary the initial segment.

Another interesting aspect of the presented examples is a product of the property of EAs to create their own biotope for the artificial life forms – in our case the street network. We can observe special strategies for the EA to maximize their fitness (the choice values): the first is to maximize the number of street segments to produce more trips and thus higher maximal choice values. This could be overcome by averaging the values i.e. dividing the choice values by the number of streets. The second is to generate street segments at strategically beneficial places (e.g. top left corner in the left and right networks in Figure 8). These segments can produce more trips via certain segments with high choice values to increase them further.

In our last example (Figure 8) we have used two goal functions: one to achieve a center and one to create a quiet area. Both are combined into one fitness function. Here we run into the problem of weighting both criteria against each other in a more or less arbitrary way. This weighting, however, has a significant effect on the optimization process und thus on the quality of the results. For example the optimization process can get stuck in local optima, since one criterion is already very good, but the other not. The improvement of poor criteria may be hindered because it may negatively affect other very good criteria, so that the resulting fitness value cannot be improved. To avoid these kind of problems we need to use evolutionary multi-criteria optimization (EMO) methods (Deb, 2001).

CONCLUSION AND OUTLOOK

In this paper we have demonstrated the potentials of using an optimization system for urban planning tasks using a test scenario. In this scenario we have generated street networks with defined local properties. The presented system is a first component of an framework with basic functionality to efficiently search compromise solutions for complex planning problems. A first software prototype has been implemented with an intuitive user interface to represent planning problems, to present various compromise solutions, and to improve them interactively.

The differences in the examples presented in Figures 6-8 show clearly that our system doesn't generate globally optimal solutions – e.g. one can delete connections that enable ring trips around the centre to increase the traffic through the centre (and to increase the corresponding choice value). This is an inherent aspect of EAs: they cannot guarantee finding the globally best solutions, but they can always offer good ones. This disadvantage can be improved by running more generations or by using separate populations in parallel and migrating the best variants between them. In our context, this isn't a problem because planners are not usually looking for global optima as goal functions represent only a part of a planning problem. Thus the interactive and adaptable search for variants is the main support for the planning process.

The next step for the development of our framework is to implement a more complex EMO system which integrates algorithms for parceling and building placement (Aliaga et al., 2008; Knecht and

Koenig, 2012). With this development we can achieve the multi-level approach illustrated in Figure 1.

ACKNOWLEDGEMENT

Special thanks go to our colleague Christian Tonn from the Bauhaus-Universität Weimar, who implemented the fast graph calculations for GPU.

REFERENCES

Aliaga, D. G., Vanegas, C. A., and Beneš, B. (2008). Interactive example-based urban layout synthesis. *Acm Transactions on Graphics, 27*(5), 1-10.

Bentley, P. J., and Corne, D. W. (2002). An Introduction to Creative Evolutionary Systems. In P. J. Bentley and D. W. Corne (Eds.), *Creative Evolutionary Systems* (pp. 1-76). San Francisco: Morgan Kaufmann.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*: John Wiley and Sons.

Derix, C. (2009). In-Between Architecture Computation. *International Journal of Architectural Computing, 7*(4).

Dillenburger, B., Braach, M., and Hovestadt, L. (2009). *Building design as an individual compromise between qualities and costs: A general approach for automated building generation under permanent cost and quality control.* Paper presented at the CAADFutures 2009.

Knecht, K., and Koenig, R. (2012). *Automatische Grundstücksumlegung mithilfe von Unterteilungsalgorithmen und typenbasierte Generierung von Stadtstrukturen*. Weimar: Bauhaus-Universität Weimar.

Koltsova, A., Tuncer, B., Georgakopoulou, S., and Schmitt, G. (2012). *Parametric tools for conceptual design support at the pedestrian urban scale*. Paper presented at the eCAADe.

Parish, Y. I. H., and Müller, P. (2001). *Procedural Modeling of Cities.* Paper presented at the SIGGRAPH, Los Angeles, CA.

Radford, A. D., and Gero, J. S. (1988). *Design by optimization in architecture, building, and construction*. New York: Van Nostrand Reinhold.

Rutton, D. (2010). Evolutionary Principles applied to Problem Solving. Retrieved 18.06.2011, from http://www.grasshopper3d.com/profiles/blogs/evolutionary-principles

Turner, A. (2001). *Angular Analysis.* Paper presented at the 3rd International Space Syntax Symposium, Atlanta.

Turner, A. (2007). From axial to road-centre lines: a new representation for space syntax and a new model of route choice for transport network analysis. *Environment and Planning B: Planning and Design, 34*(3), 539 – 555.

[1] http://www.grasshopper3d.com/ (Retrieved 01.02.2013)

[2] http://www.bentley.com/en-US/Promo/Generative%20Components/default.htm

[3] http://www.esri.com/software/cityengine (Retrieved 01.02.2013)

[4] Galapagos is a plugin for evolutionary optimization for Grasshopper/Rhino3D: http://www.grasshopper3d.com/group/galapagos (Retrieved 01.02.2013)

[5] http://en.wikipedia.org/wiki/Inverse_problem (Retrieved 01.02.2013)

[6] Bundesamt für Landestopografie, swisstopo (Art. 30 GeoIV), © 2011 swisstopo

[7] http://www.aforgenet.com/ (Retrieved 21.05.2013)

[8]http://www.shodor.org/petascale/materials/UPModules/dynamicProgrammingPartI